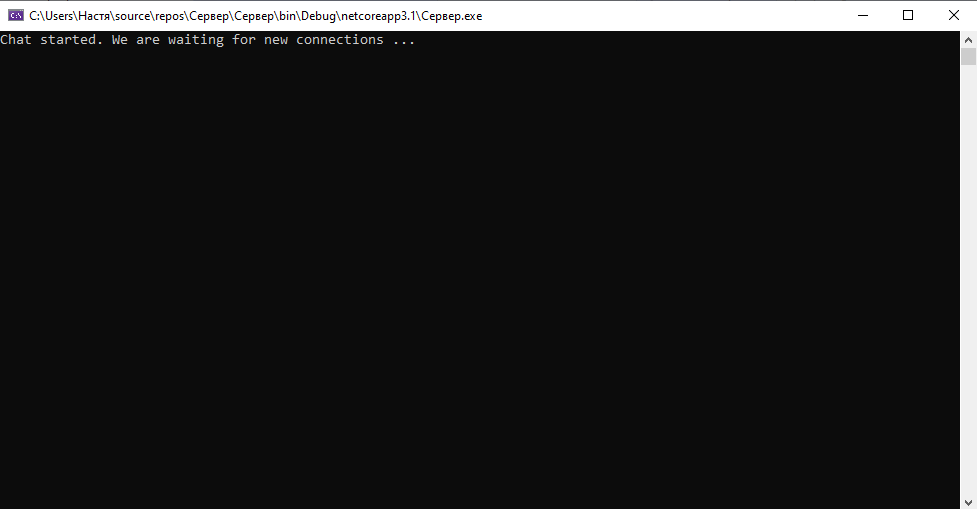
Кузнецова Анастасия 381707-2

# Отчёт по лабораторной работе: «Чат».

Содержание работы:

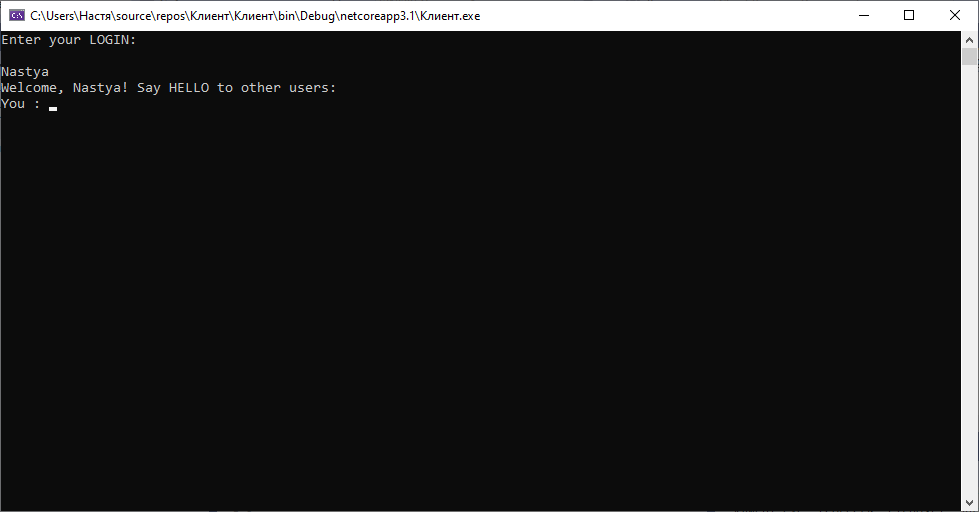
Общение нескольких клиентов, имеющих собственное имя(логин), через общий сервер, выполняемое по средствам Socket.

1. **Запуск сервера:**

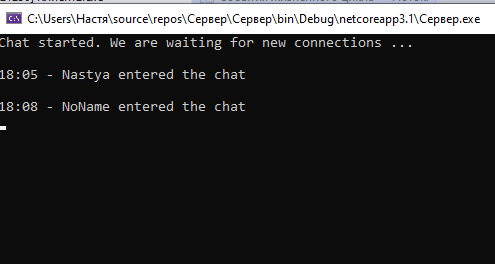


1. **Запуск Клиента:**

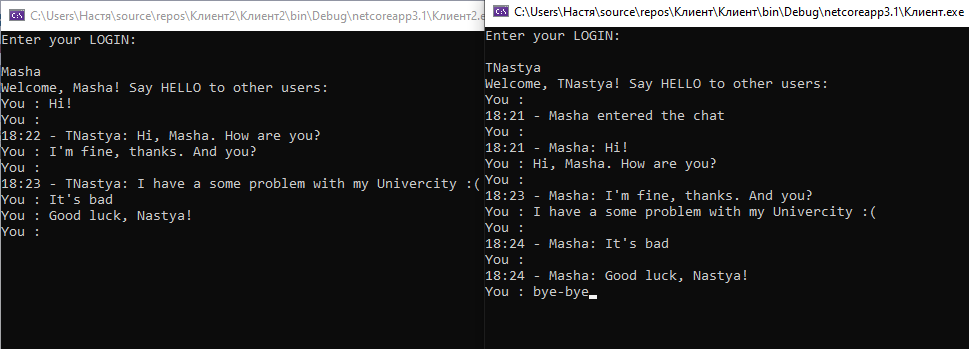
При запуске, консоль просит указать персональное имя пользователя, чтобы в дальнейшем обращаться к нему по имени(логину).

****

1. **При вхождении пользователей, на консольном окне сервера и на окне остальных пользователей отобразится оповещение с указанием времени.**

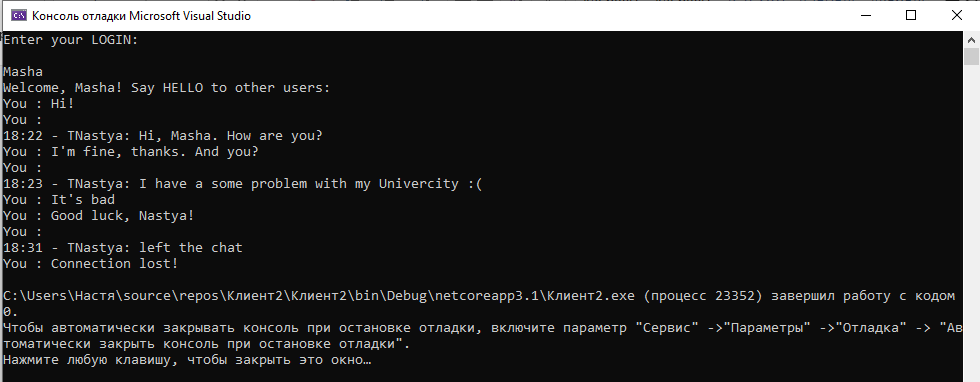
****

1. **Каждый пользователь может писать любое сообщение. И его сообщение отобразится как на консоли у сервера, так и на консоли у остальных пользователей.**

****

1. **- Когда пользователь выходит из чата, это отображается на консоли у остальных пользователей и у сервера.**

**- Когда сервер отключается, нам выдается сообщение о том, что соединение потеряна.**

****

**Примечание:**

При создании программы я взяла структуру из этого источника:

<https://metanit.com/sharp/net/4.4.php>

Адаптировала его под Socket с помощью видеоуроков:

<https://www.youtube.com/watch?v=ZRGgBtUgJKE&t=1298s>

# Приложение

В сервере реализовано два класса: ClientObject и ServerObject

ClientObject:

1. **using** System;
2. **using** System.Text;
3. **using** System.Net;
4. **using** System.Net.Sockets;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Threading;
9. **namespace** Сервер
10. {
11. **public** **class** ClientObject
12. {
13. **protected** **internal** **string** Id
14. {
15. **get**;
16. **private** **set**;
17. }
18. **string** userName;
20. **protected** **internal** Socket client;
21. ServerObject server;
23. **public** ClientObject(Socket socket, ServerObject serverObject)
24. {
25. Id = Guid.NewGuid().ToString();
26. client = socket;
27. server = serverObject;
28. serverObject.AddConnection(**this**);
29. }
31. **public** **void** Process()
32. {
33. **try**
34. {
35. **string** message = GetMessage(client);
36. userName = message;
37. message = "\n" + DateTime.Now.ToShortTimeString() + " - " + userName + " entered the chat";
38. server.BroadcastMessage(message, **this**.Id);
39. Console.WriteLine(message);
40. **while** (**true**)
41. {
42. **try**
43. {
44. **string** get\_message = GetMessage(client);
45. message = String.Format("\n{0} - {1}: {2}", DateTime.Now.ToShortTimeString(), userName, get\_message); Console.WriteLine(message);
46. server.BroadcastMessage(message, **this**.Id);
47. **if** (get\_message == "!quit")
48. {
49. message = String.Format("\n{0} - {1}: left the chat", DateTime.Now.ToShortTimeString(), userName);
50. Console.WriteLine(message);
51. server.BroadcastMessage(message, **this**.Id);
52. **break**;
53. }
54. }
55. **catch**
56. {
57. message = String.Format("\n{0} - {1}: left the chat", DateTime.Now.ToShortTimeString(), userName);
58. Console.WriteLine(message);
59. server.BroadcastMessage(message, **this**.Id);
60. **break**;
61. }
62. }
63. }
64. **catch** (Exception e)
65. {
66. Console.WriteLine(e.Message);
67. }
68. **finally**
69. {
70. server.RemoveConnection(**this**.Id);
71. Close(client);
72. }
73. }
75. **private** **string** GetMessage(Socket handler)
76. {
78. **byte**[] data = **new** **byte**[64];
79. StringBuilder builder = **new** StringBuilder();
80. **int** bytes = 0;
81. **do**
82. {
83. bytes = handler.Receive(data);
84. builder.Append(Encoding.Unicode.GetString(data, 0, bytes));
85. }
86. **while** (handler.Available > 0);
87. **return** builder.ToString();
88. }
89. **protected** **internal** **void** Close(Socket handler)
90. {
91. handler.Shutdown(SocketShutdown.Both);
92. handler.Close();
93. }
94. }
95. }

ServerObject:

1. **using** System;
2. **using** System.Text;
3. **using** System.Net;
4. **using** System.Net.Sockets;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Threading;

10. **namespace** Сервер
11. {
12. **public** **class** ServerObject
13. {
14. **static** **int** port = 8888;
15. **static** Socket Soc\_Listener;
16. List<ClientObject> clients = **new** List<ClientObject>();
17. **protected** **internal** **void** AddConnection(ClientObject clientObject)
18. {
19. clients.Add(clientObject);
20. }
22. **protected** **internal** **void** RemoveConnection(**string** id)
23. {
24. ClientObject client = clients.FirstOrDefault(c => c.Id == id);
25. **if** (client != **null**)
26. clients.Remove(client);
27. }
29. **protected** **internal** **void** Listen()
30. {
31. **try**
32. {
33. IPEndPoint ipPoint = **new** IPEndPoint(IPAddress.Parse("127.0.0.1"), port);
34. Soc\_Listener = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
35. Soc\_Listener.Bind(ipPoint);
36. Console.WriteLine("Chat started. We are waiting for new connections ...");
37. **while** (**true**)
38. {
39. Soc\_Listener.Listen(10);
40. Socket handler = Soc\_Listener.Accept();
41. ClientObject clientObject = **new** ClientObject(handler, **this**);
42. Thread clientThread = **new** Thread(**new** ThreadStart(clientObject.Process));
43. clientThread.Start();
44. }
45. }
46. **catch** (Exception ex)
47. {
48. Console.WriteLine(ex.Message);
49. Disconnect();
50. }
51. }
53. **protected** **internal** **void** BroadcastMessage(**string** message, **string** id)
54. {
55. **byte**[] data = Encoding.Unicode.GetBytes(message);
56. **for** (**int** i = 0; i < clients.Count; i++)
57. {
58. **if** (clients[i].Id != id)
59. {
60. clients[i].client.Send(data);
61. }
62. }
63. }
65. **protected** **internal** **void** Disconnect()
66. {
67. Soc\_Listener.Close();
68. **for** (**int** i = 0; i < clients.Count; i++)
69. {
70. clients[i].client.Close();
71. }
72. Environment.Exit(0);
73. }
74. }
75. }

Сам Сервер:

1. **using** System;
2. **using** System.Text;
3. **using** System.Net;
4. **using** System.Net.Sockets;
5. **using** System.Collections.Generic;
6. **using** System.Linq;
7. **using** System.Threading;
9. **namespace** Сервер
10. {
11. **internal** **class** Program
12. {
13. **static** ServerObject server;
14. **static** Thread listenThread;
15. **static** **void** Main()
16. {
17. **try**
18. {
19. server = **new** ServerObject();
20. listenThread = **new** Thread(**new** ThreadStart(server.Listen));
21. listenThread.Start();
22. }
23. **catch** (Exception ex)
24. {
25. server.Disconnect();
26. Console.WriteLine(ex.Message);
27. }
28. }
29. }
30. }

Клиент:

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
5. **using** System.Threading.Tasks;
6. **using** System.Net;
7. **using** System.Net.Sockets;
8. **using** System.Threading;
10. **namespace** Client
11. {
12. **class** Program
13. {
14. **static** **string** userName;
15. **static** **int** port = 8888; // порт сервера
16. **static** **string** address = "127.0.0.1"; // адрес сервера
18. **static** **void** SendMessage(Socket socket)
19. {
20. **while** (**true**)
21. {
22. Console.Write("You : ");
23. **string** message = Console.ReadLine();
24. **byte**[] data = Encoding.Unicode.GetBytes(message);
25. **while** (message == "")
26. {
27. message = Console.ReadLine();
28. data = Encoding.Unicode.GetBytes(message);
29. }
30. socket.Send(data);
31. **if** (message == "!quit")
32. {
33. Disconnect(socket);
34. **break**;
35. }
36. }
37. }
39. **static** **void** ReceiveMessage(**object** obj)
40. {
41. Socket socket = (Socket)obj; **while** (**true**)
42. {
43. **try**
44. {
45. **byte**[] data = **new** **byte**[256];
46. StringBuilder builder = **new** StringBuilder();
47. **int** bytes = 0;
48. **do**
49. { bytes = socket.Receive(data, data.Length, 0);
50. builder.Append(Encoding.Unicode.GetString(data, 0, bytes));
51. }
52. **while** (socket.Available > 0);
53. **string** message = builder.ToString();
54. //string message1 = String.Format("\n{0} : {1}", DateTime.Now.ToShortTimeString(), message);
55. Console.WriteLine(message);
56. Console.Write("You : ");
57. }
58. **catch**
59. {
60. Console.WriteLine("Connection lost!");
62. Disconnect(socket);
63. }
64. }
65. }
67. **static** **void** Disconnect(**object** obj)
68. {
69. Socket socket = (Socket)obj;
70. socket.Shutdown(SocketShutdown.Both);
71. socket.Close();
72. Environment.Exit(0);
73. }


77. **static** **void** Main(**string**[] args)
78. {
79. Console.WriteLine("Enter your LOGIN:\n");
80. userName = Console.ReadLine();
82. IPEndPoint ipPoint = **new** IPEndPoint(IPAddress.Parse(address), port);
83. Socket socket = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
84. **try**
85. {
86. socket.Connect(ipPoint);
87. **string** message = userName;
88. **byte**[] data = Encoding.Unicode.GetBytes(message);
89. socket.Send(data);
90. Thread receiveThread = **new** Thread(ReceiveMessage);
91. receiveThread.Start(socket);
92. Console.WriteLine("Welcome, {0}", userName+ "! Say HELLO to other users:");
93. SendMessage(socket);
94. }
95. **catch** (Exception ex)
96. {
97. Console.WriteLine(ex.Message);
98. }
99. **finally**
100. {
101. Disconnect(socket);
102. }
103. }
105. }
106. }